Augmented Computing at the Edge Using Named Data Networking

Rustam Pirmagomedov*, Srikathyayani Srikanteswara†, Dmitri Moltchanov*, Gabriel Arrobo†,
Yi Zhang†, Nageen Himayat†, and Yevgeni Koucheryavy*
*Unit of Electrical Engineering, Tampere University, Finland
†Intel Corporation, USA

{rustam.pirmagomedov, dmitri.moltchanov, evgeny.kucheryavy}@tuni.fi
{srikathyayani.srikanteswara, gabriel.arrobo, yi1.zhang, nageen.himayat}@intel.com

***Abstract*—Edge computing is considered vital to IoT evolution, enabling the timely execution of various computational tasks for constrained devices utilizing external resources. The conventional host-based network architectures become a bottleneck for further development of edge computing, primarily when serving latency-sensitive applications. Further, existing approaches do not exploit complex data correlations in the network layer for optimization. This paper demonstrates that Named Data Networking (NDN) has the potential to enable efficient support for mobile users offloading their time-sensitive computing tasks to edge servers. For this purpose, the NDN protocol was enhanced with a server selection procedure, capable of adjusting for the varying resource availability on edge servers. The results of the experiments show clear support for using NDN in these scenarios, with individual gains coming not just from Interest aggregation and caching, which are NDN features, but also from dynamic server selection.**

***Index Terms*—ICN, NDN, edge computing, Future Internet**

I. Introduction

The ubiquitous proliferation of TCP/IP networking solutions over the last few decades has gradually led to the existing Internet architecture. This architecture, while still capable of supporting modern applications, faces significant challenges from emerging services and applications that are more distributed and should operate in a dynamically changing environment. As a result, recently, there have been several initiatives aimed at rethinking the foundation of network design principles. Although this trend is accompanied by the industrys inherent inertia for improvement, there is growing evidence that a plethora of emerging services and use-cases may benefit from these new networking approaches.

The Named Data Networking (NDN) paradigm promotes the idea of constructing the future internet around the data rather than hosts [1]. In the NDN framework, data is requested by sending interests, which effectively eliminates the need for hostname resolution. Moreover, NDN supports the aggregation of interests, allowing for better efficiency of network resource utilization in case the requested content overlaps. Finally, NDN natively allows for local caching of requested data, which improves tolerance to link failures and topology changing.

Conventionally, NDN is considered to replace the functionality of the network and transport layers. However, NDN may also enhance the performance at the *service layer*, especially for services associated with dynamically changing resources and demands. In particular, NDN may be applied to dynamic service deployment at the edge, where service designers face a complex trade-off between application requirements, resource capabilities, and the inherently dynamic network topology [2]. Initial studies in this topic have shown that NDN technologies hold the potential for orchestrating computation and efficient re-use of previously computed results [3]. However, those works are lacking efficient solutions for dynamic server selection, and do not elaborate on the numerical gains enabled by NDN technology for the dynamic service provisioning.

In this paper, augmented computing at the edge is enabled using NDN for mobile wireless access networks. Particularly, it is considered that mobile users offload compute tasks to the edge, with dynamic server selection based on their current load. The developed solution does not interfere with default NDN functionality but complements it with a service discovery procedure. The main contributions of this paper are as follows:

- An *augmented computing paradigm* that can seamlessly increase the computing capabilities available on a mobile device by using third-party resources (e.g., edge or cloud servers) over NDN;
- A *service discovery procedure* to enhance the NDN protocol that allows receiving offers (multiple Data packets) from different service providers back to a single service request (Interest packet) broadcasted to the network;
- A method for *delegated edge server selection* over NDN.

The remainder of the paper is organized as follows: motivation and a use-case description are provided in Section II. Further, in Section III, the improvements made to the NDN protocol and a proposed method for edge server selection are described. In Section IV, computer simulations are presented and discussed; conclusions are drawn in Section V.

II. Motivation

A. The Considered Use-Case: Dynamic Compute Service

This work considers a dynamic compute service, where each computing task is defined by data that needs to be processed and software used for the processing of the data. For example,

a compute task for 3D rendering may be composed of photos of an object (data) and the software required for processing these photos. Each mobile device has a queue of computing tasks to be performed. Assuming that the complexity of each task in the queue is known, the user device can be capable of estimating when the CPU will begin processing each of the tasks.

Further, it is assumed that a remote computing infrastructure (edge or cloud) is available for mobile devices to use. A user/mobile device opportunistically offloads some tasks from its queue to a remote server. If the results are delivered back before the processing of that task is scheduled in the user device, the task is considered as accomplished and is removed from the queue. This target time, T_D, is referred to as the requested service accomplishing time.

The rationale in considering NDN for this use-case relies on the common assumption that computing requests issued by users are likely to be correlated [3]. For example, in the VANET scenario, cars are reacting to a situation (e.g., congestion or an accident on the road) using the same data (e.g., data disseminating from the vehicle that first detected the accident) with only a limited number of software options for processing that data. Thus, users requesting the same processing operations over the same data may potentially benefit from NDN capabilities coming from interest aggregation and caching. These gains increase with higher similarity among user requests. However, compared to previous studies, here we demonstrate that NDN can enhance performance not only via exploiting similarities of the requested services but also via handling the dynamism of available resources (e.g., CPU load at edge servers).

B. Challenges of Edge Computing

Edge computing is growing exponentially through the increase of sensors, data, and computing being performed at the edge. In many emerging dynamic edge compute applications, low latency is needed even in the face of changing network structures, varying wireless links, and resources in the presence of mobility. Additionally, time-sensitive analysis entails constructing the right services dynamically with the right data in real-time. In such situations, it is entirely possible that the source of the data or the destination (the best place to perform the compute) may not be known a priori. Further, existing IP-based solutions do not have any awareness of the network layer to exploit similarities in data or compute requests. Since emerging edge applications can have a rich correlation in compute requests and data (as discussed in the Section II-A), name-based networking solutions allow the exploitation of these correlations resulting in the more efficient solution.

Recent works in using NDN for computing, focus on placing functions in the network, and executing them [4]. Named Function Networking (NFN) uses ICN naming to specify the compute function to be performed (code) and data to be processed [5]. In addition to enabling a remote procedure call (RPC), NFN includes methods for fetching results that exist somewhere in the network. RICE [6], further extended NFN functionality with consumer authentication and authorization, enhanced service parameter passing mechanism, and accommodating non-trivial (nested) computations. RPC functionalities, with the focus on the edge and fog computing, are extended in NFaaS [7]. If the first compute server that receives the request does not have enough resources (e.g., CPU) to execute the task, it forwards the request to a neighboring node. Such an approach provides opportunistic accomplishment of the compute task but does not guarantee either the best server selection or the satisfaction of time constraints for latency-sensitive services.

The server selection procedure represents a significant challenge for edge and fog computing [8]. Compute resources available at the edge servers often change rapidly, and this should be taken into account when selecting a server for offloading a computing task. Proactive approaches (e.g., NLSR [routing]), which rely on periodic updates of the network state, are not suitable for dynamic environments due to high signaling overhead. At the same time, reactive approaches introduce additional latency and may not address the demand for latency-sensitive applications. In [9], the authors proposed to collect offers from computing-capable nodes when fetching data to be processed. This approach introduces minimal overhead for the selection procedure (in terms of both selection time and signaling) and guarantees the best server selection among those located between a consumer and the location of the data. However, this approach does not consider servers that are in other directions and can provide better offers.

In this work, addressing the challenge of dynamic server selection at the edge for serving latency-sensitive computing requests is the focus.

III. Dynamic Server Selection in NDN

A. Leveraging Multiple Responses

In NDN, an interest packet is sent to request a specific piece of data identified by its name. Nodes that have the requested content reply with a data packet. However, only the first data packet received by intermediary nodes is relayed back to the node that sent the interest packet. This efficiently reduces the network load when delivering data. On the other hand, enabling the delivery of multiple data packets per one interest also can be justified for specific scenarios such as when a user wants to receive offers from multiple service providers. Modifying the paradigm of "one data packet per interest" to enable dynamic compute server selection on the edge for compute-related names is explored in this paper.

To enable multiple data packets per one interest, a "Discover" type of interest packet is introduced. To achieve this, a specially allocated keyword is used, which can be embedded into the namespace. For example, a namespace may start with a keyword prefix such as "/discover/service/". In other words, certain predefined and known keywords are reserved to obtain multiple data packets in response to a single interest packet. Per this design, the reserved keywords cannot be assigned as the root of any other namespace. Moreover, it is also targeted as a healthy balance between overhead in the

network and diversity gain related to server selection. This is achieved by limiting the number of Data packets relayed by intermediary nodes based on both a timer and the number of times the Data packet has been forwarded downstream. Every service discovery interest contains the desired number of responses the consumer wants to receive back along with the "InterestLifetime" field. These two elements are used to decide when the entry for this interest can be removed from the Pending Interest Table (PIT).

To prevent stale entries in the PIT table, routers along the path of the "Discover" packet compare these parameters with their own policies. If the parameters are policy-compliant, the parameters can be applied to the appropriate PIT entry. Otherwise, the number of responses can be decreased by a router according to its internal policy. Once the "InterestLifetime" times out, the PIT entry will be deleted if no responses have been received. When the data comes back, and the associated PIT entry has not yet expired, the forwarder will satisfy the interest by sending the data packet downstream and decreasing the counter by one. Once the counter reaches zero, implying that the number of the packets sent back reached the requested number of responses, its associated PIT entry is marked as expired and deleted.

B. Server Selection by Node Delegation

In a wireless access environment, especially in the unlicensed spectrum, the interference and high mobility of users may lead to performance degradation of the considered dynamic compute service. To alleviate the problem of server selection, it is proposed to use a delegated node in the wired part of the network on behalf of a mobile user. Such an approach allows for shifting the signaling overhead of the selection procedure from the wireless links to the more reliable wired infrastructure. Additionally, it reduces the risk of interruptions during the selection procedure and enhances stability. The procedure for server selection is presented in Fig. 1.

The procedure starts with a request (Interest) for a service sent by a mobile user (1). The Interest includes details of the requested service inserted in the namespace. The namespace of the Interest (1) starts with a keyword specifying that delegated server selection is demanded. After receiving the Interest, a node capable of delegated server selection sends "Discover" type of Interest packets (2.1, 2.2) to the network and waits for offers from available servers. Servers receiving the "Discover" interest packet, reply back with the cost offer (3.1, 3.2). The offer may include the expected computation time, the price for the service, server identifier, and any other relevant metrics. Note that the delegated node may receive multiple offers from servers for the same Interest, as described in Section III-A.

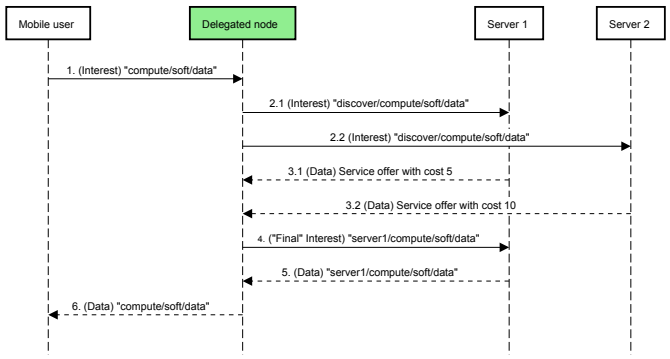


Fig. 1. Server selection by a delegated node.

It is worth noting that Servers may not have the data or software needed for the requested computation. Thus, when making an offer, a server should estimate and take into account the time and costs required for acquiring the data and software specified in the request. Such estimation can be actively made by sending a request to the origins, or passively relying on the information in routing or forwarding information bases (RIB or FIB, respectively). The pros and cons of these two approaches are outside of the scope of this paper.

Upon reception of the first offer, the delegated node activates a decision timer. Before the timer expires, the delegated node may receive offers from multiple servers. After the decision timer expires, the delegated node sends the "final" Interest (4) to the server that provided the best offer. The "final" Interest contains a server identifier in the namespace, thus only the selected server may reply for such Interest. It is worth noting that the "final" Interest can be sent even if the timer has not expired; for instance if the predefined number of offers have been received. The mobile user provides information about the demanded number of offers from servers and decision timer duration as part of the Interest in (1). The "final" Interest received by a server is considered as a confirmation that this server was selected for performing the task. Once computing is done by the server, it replies to the "final" Interest with a data packet(s) containing the results (5). Having received the results of the computation, the delegated node optionally stores it in its cache for future reuse and delivers the data to the user (6).

It is worth noting that the proposed procedure is meant for latency-sensitive applications (i.e., fast computations requiring a result delivery time much lower than one second). If the proposed method is used for long-time computations, the lifetime of Interests (1) and (4) should be adjusted accordingly.

IV. Performance Evaluation

A. Simulation Setup

To assess the performance of NDN for augmented computing applications, a highway scenario is considered where mobile users move along and utilize a cloud and edge servers for offloading computation tasks. To assess the response of the system, light and congested traffic conditions are modelled. The light traffic conditions assume the mean density of traffic is about 10 vehicles per kilometer, and the average velocity of 40 m/s. The congested traffic conditions imply the density of 40 vehicles per kilometer, and velocity set to 10 m/s.

Connectivity is enabled by wireless small cells deployed along the road. To conduct the performance evaluation, NDN

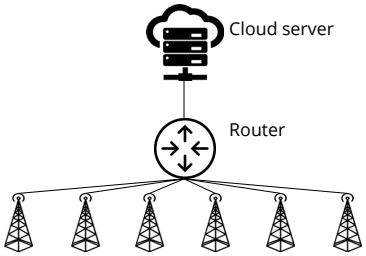


Fig. 2. Cloud-centric deployment

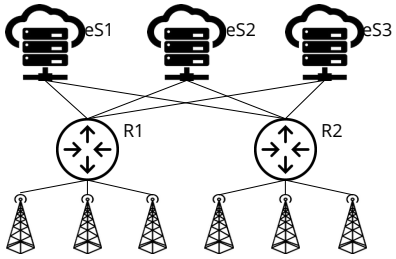


Fig. 3. Edge-centric deployment; R1, R2 - routers, eS1-eS3 edge servers

community simulator (ndnSIM) [10] was extended with the features described in Section III.

Cloud-centric (Fig. 2) and edge-centric (Fig. 3) deployments were tested. These deployments rely on *unlicensed* wireless technologies and do not assume the availability of any handover procedure typically available in cellular networks (e.g., 5G). This is exceptionally challenging for latency-sensitive applications due to the potential loss of packets in the wireless interface.

TABLE I
PARAMETERS FOR SIMULATION ASSESSMENT

Parameter	**Value**
Number of runs for each combination of parameters	100
Number of users in light traffic conditions, N per km	10
Number of users in congested traffic conditions, N per km	40
Velocity of users in light traffic conditions, Vm/s	40
Velocity of users in congested traffic conditions, Vm/s	10
NDN application types	cbr, zipf
IP application type	UDP
Cache size, number of compute results	0, 20
Cache type	LRU
Computation results freshness, ms	1000
Compute time distribution, ms	U [10;100]
Decision time for server selection, ms	6
Mean RTT between a user and a cloud, ms	100
Mean RTT between a user and edge servers, ms	[3;6]

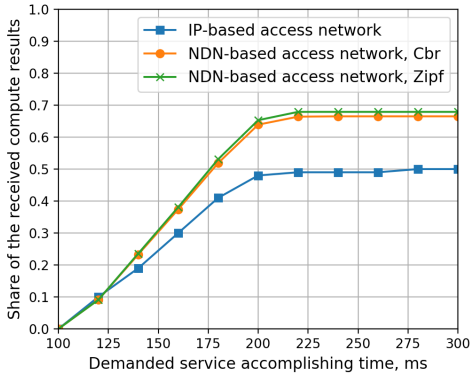
It is assumed that services requested by mobile users may overlap, which means that users may request the same processing operations over the same data. This assumption is motivated by a typical highway scenario where autonomous vehicles react to an event (e.g., traffic accident or changing road conditions), utilizing standard processing algorithms (types of software). Two types of applications, "cbr" and "zipf", are used. In the former case, the simulation setup reflects users requesting the same services (using the same software to process the event).

In the case of the "zipf" application, it is modelled a situation where users may utilize different software for processing the data limiting the total number of software options to 100. Moreover, some applications/services are more popular than others, with the popularity distributed according to the ZipfMandelbrot law. In this case, mobile users may request different processing services when reacting to road events. However, the requests may still overlap due to the popularity of some software. In both applications, the freshness of data is limited to one second. This means that data processing results are treated as expired after one second and deleted from cache (when caching is enabled).

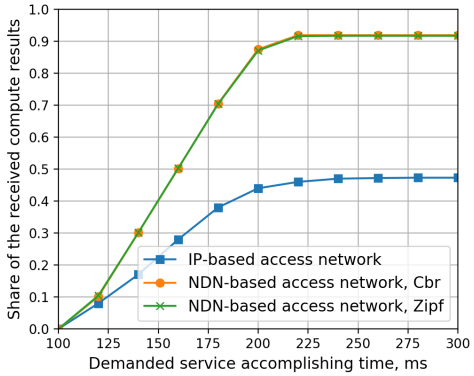
The performance metric used is the success rate of computing request accomplished on time. This metric integrates both latency and reliability in a single bundle providing comprehensive insights on the system behavior. The main simulation parameters are summarized in Table I.

B. The Effect of Interest Aggregation

In the first set of experiments, an IP-based solution was characterized, assuming a user device offloads computation tasks to the cloud over the User Datagram Protocol (UDP). The UDP is a typical choice for latency-sensitive applications similar to the considered in this work (described in Section II-A).



(a) Free road

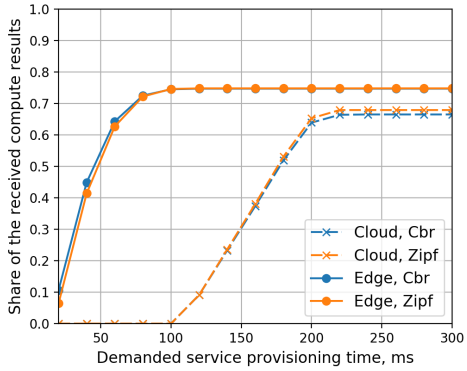


(b) Congested road

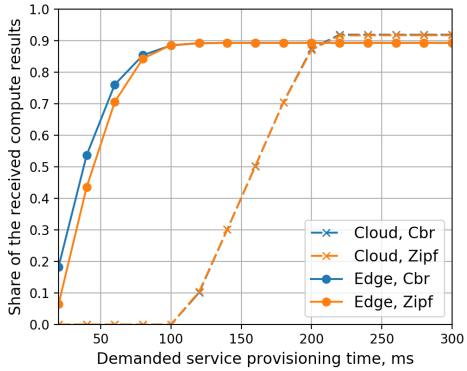
Fig. 4. The effect of interests aggregation (caching disabled)

In addition to the IP-based deployment, an IP cloud service combined with the NDN-based access is considered. In this deployment, the access points and the router in Fig. 2 operate over NDN, while the cloud server operate over IP. The router serves as a gateway between the NDN and IP networks. Such a combination, supplemented with disabled caching, allows characterizing the benefits of Interest aggregation provided by NDN. The results are presented in Fig. 4.

The homogeneous IP stack reaches about 50% of successfully accomplished computing requests within the first 300 ms, for both light and congested traffic conditions. This implies that the challenges of high mobility associated with the light traffic conditions have almost the same contribution to the losses as the increased interference and network load in a congested traffic case. Moreover, note that the interest aggregation feature available in NDN technology allows for a notable increase in the share of the computing requests accomplished on time for both zipf and cbr applications. The superior performance showed by NDN can be explained not only by its interest aggregation but also due to the overhead of resolving host addresses in IP-based deployment. Additionally, the improvements for the congested road traffic scenario outperform those for the light traffic conditions because of the higher overlap in the services requested by mobile users.

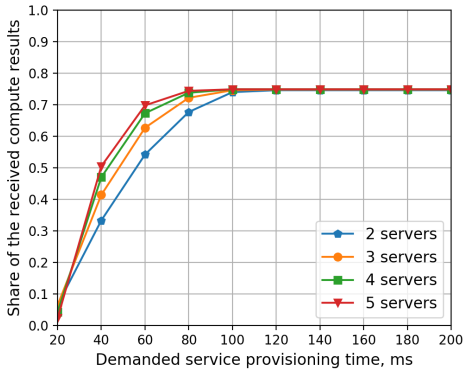


(a) Free road

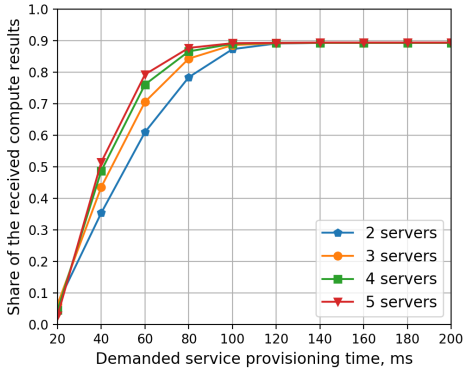


(b) Congested road

Fig. 5. Comparison of the edge and cloud deployments (caching disabled)



(a) Free road



(b) Congested road

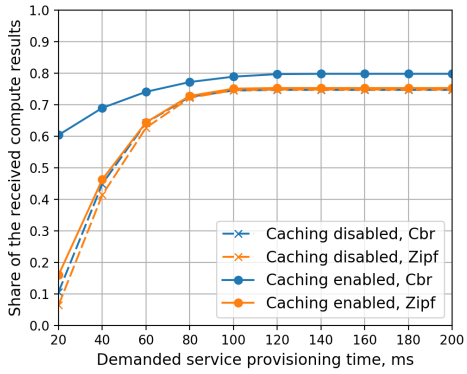
Fig. 6. The effect of server selection (caching disabled, Zipf application)

C. The Effect of Server Selection

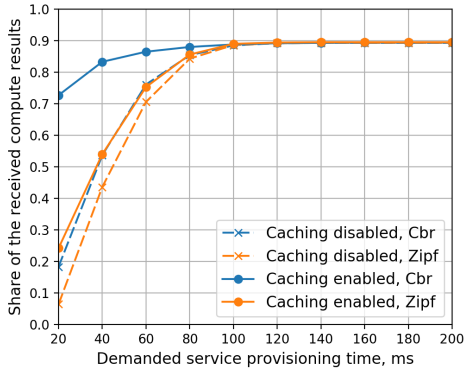
The gains of the edge computing with dynamic server selection is evaluated in this subsection. For this set of experiments, the deployment used is illustrated in Fig. 3. In this scenario, routers R1 and R2 perform server selection on behalf of mobile users, following the methodology described in Section III.

The results presented in Fig. 5 compare the performance of the NDN-based edge and the cloud-centric deployments. The edge-centric scenario provides significantly lower service accomplishing time, and rapid growth of the served computing requests. The reason for this effect is not only the reduced latency between mobile users and computing servers but also the overall increase in the number of available computing resources. Having three servers, it is more likely that a faster option is available for each of the compute requests.

When servers are hosted at the edge, the effect of interest aggregation is reduced compared to a cloud-centric deployment. In the case of edge deployment, interests from three access points (of the same area) are aggregated, while in the case of cloud-centric implementation, aggregation covers all the access points (i.e., six access points). This effect results in a slightly higher number of served compute requests in the congested scenario (Fig. 5(b)). In the free road scenario, reduced service time allows for a reduction of losses caused by high mobility, resulting in a higher number of the request served within 300 ms as compared to the cloud-centric scenario, even with reduced efficiency of interest aggregation (Fig. 5(a)).



(a) Free road



(b) Congested road

Fig. 7. The effect of caching

To better highlight the effect of server selection on the edge, deployments with a different number of servers were simulated (Fig. 6). The results of the experiments show that a higher number of servers with the same compute time distribution results in lower mean computation time per request, which in turn, leads to a higher share of the computing services accomplished within a shorter time. These results confirm that the proposed method can balance the load on the servers dynamically, depending on their current availability.

D. The Effect of Caching

The final set of experiments present the performance improvements to the edge scenario due to caching. Caching is enabled in both routers (R1, R2) and all access points. The cache allows for the storage of at most 20 results for no more than 1000 ms as specified in Table I. The considered deployment is shown in Fig. 3 and the comparison between cache enabled and cache disabled for the edge scenario is shown in Fig. 7.

It is observed that caching allows for significant improvement in the fraction of service requests completed in time for "cbr" traffic pattern. The "cbr" users request the same service within the 1000 ms (inter-request time for the used frequency of interest). Provided that freshness of the computation results is set to 1000 ms, the Interests for the same service from other users will be satisfied with the results from the cache, where it was stored after the first request. Differently, in the case of "zipf" application, vehicles may react to the same event (data) using different software. As a result, only a limited number of requests overlap in time, reducing the gains of caching. Therefore, for latency-sensitive applications, the effect of caching is limited to very specific use-cases.

V. Conclusions

Addressing the augmented computing paradigm, in this paper, a novel method for dynamic computing over NDN is proposed and evaluated. The benefits of using NDN for augmented computing were demonstrated using NDN in wireless access networks. The simulation results show the benefits achieved by each of the considered NDN features independently (interest aggregation and caching) and by the dynamic server selection method.

The gains of interest aggregation and caching are very application-specific and may improve the performance by several magnitudes if the services requested by users significantly overlap in time. In contrast, the proposed server selection method allows for balancing the load among servers, regardless of overlap in service requests.

In summary, this research has shown that NDN-based solutions may successfully handle the dynamics of resource availability on the edge providing notable benefits for compute services.

References

[1] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, p. 6673, Jul. 2014. [Online]. Available: https://doi.org/10.1145/2656877.2656887

[2] P. Simoens, D. Griffin, E. Maini, T. K. Phan, M. Rio, L. Vermoesen, F. Vandeputte, F. Schamel, and D. Burstzynowski, "Service-centric networking for distributed heterogeneous clouds," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 208–215, 2017.

[3] M. Sifalakis, B. Kohler, C. Scherb, and C. Tschudin, "An information centric network for computing the distribution of computations," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ser. ACM-ICN 14. New York, NY, USA: Association for Computing Machinery, 2014, p. 137146. [Online]. Available: https://doi.org/10.1145/2660129.2660150

[4] R. Tourani, S. Srikanteswara, S. Misra, R. Chow, L. Yang, X. Liu, and Y. Zhang, "Democratizing the edge: A pervasive edge computing framework," *arXiv preprint arXiv:2007.00641*, 2020.

[5] C. Tschudin and M. Sifalakis, "Named functions and cached computations," in *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, 2014, pp. 851–857.

[6] M. Król, K. Habak, D. Oran, D. Kutscher, and I. Psaras, "RICE: Remote method invocation in ICN," in *Proceedings of the 5th ACM Conference on Information-Centric Networking*, 2018, pp. 1–11.

[7] M. Król and I. Psaras, "Nfaas: named function as a service," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*, 2017, pp. 134–144.

[8] A. Mtibaa, R. Tourani, S. Misra, J. Burke, and L. Zhang, "Towards edge computing over named data networking," in *2018 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 2018, pp. 117–120.

[9] M. Amadeo, G. Ruggeri, C. Campolo, and A. Molinaro, "Iot services allocation at the edge via named data networking: From optimal bounds to practical design," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 661–674, 2019.

[10] "ndnSIM Documentation." [Online]. Available: https://ndnsim.net/2.7/